Tribhuvan University

Faculty of Humanities and Social Sciences

PROJECT REPORT ON JOB PORTAL Using Naive Bayes Algorithm

Submitted to Department of ICT

Kathmandu Shiksha Campus

In partial fulfillment of the requirements for the Bachelors in Computer Application

Submitted by:

Laxman Rumba

Under the Supervision of

Akhilesh Yadav

Tribhuvan University

Faculty of Humanities and Social Sciences

Kathmandu Shiksha Campus

Supervisor's Recommendation

I hereby recommend that this project prepared under my supervision by Laxman Rumba entitled "**JOB PORTAL**" in partial fulfillment of the requirements for the degree of Bachelor of Computer Application is recommended for the final evaluation.

____

Akhilesh Yadav

BCA

Department of ICT

Kathmandu Shiksha Campus, Satungal

Tribhuvan University

Faculty of Humanities and Social Sciences

Kathmandu Shiksha Campus

LETTER OF APPROVAL

This is to certify that this project prepared by Laxman Rumba entitled “**JOB PORTAL”** in partial fulfillment of the requirements for the degree of Bachelor in Computer Application has been evaluated. In our opinion it is satisfactory in the scope and quality as a project for the required degree.

SIGNATURE of Supervisor ____________________ Akhilesh Yadav, Supervisor Kathmandu Shiksha Campus, Satungal	SIGNATURE of HOD/ Coordinator ____________________ Yuba Raj Devkota, ICT, HOD Kathmandu Shiksha Campus, Satungal
SIGNATURE of Internal Examiner ____________________ Internal Examiner	SIGNATURE of External Examiner ____________________ External Examiner

KATHMANDU SHIKSHA CAMPUS

Subject: Approval of Project Proposal

The project entitled "**JOB PORTAL**" proposed by Laxman Rumba for the partial fulfilment of the requirement for Bachelor in Computer Application (BCA), sixth semester has been approved for further development.

<u>Proposal Evaluation Committee</u>

1. ____________________________

2. ____________________________

3. ____________________________

4. ____________________________

…………………………….

Mr. Shatrughan Prasad Gupta

Campus Chief

DECLARATION

We, hereby declare that the project entitled "**JOB PORTAL**" is the best on our knowledge this is our original work, and it has not been submitted for the candidature of ours intern report to any university, college or educational institutions. It is quite possible that there are errors of omission and commission in this mostly single-handed attempt.
The project covers a never-before opportunity to the reader to acquire depth knowledge of Job Portal**.**

ABSTRACT

This Job Portal website is a web-based application developed using the MERN (MongoDB, Express.js, React, Node.js) stack, aimed at streamlining the job search and recruitment process for job seekers, recruiters, and administrators. The platform leverages intelligent algorithms such as Naive Bayes for personalized job recommendations based on the user's uploaded resume.

The system provides secure JWT-based authentication and OTP-based registration for both recruiters and job seekers. Recruiters can post jobs, manage applicants, and control job visibility, while job seekers can apply for jobs, track their applications, and receive AI-generated career insights, including salary trends in their preferred industries. Additionally, an AI-powered quiz feature helps users assess their skills and guides them toward suitable career paths.

For administrative users, the system includes a dedicated dashboard to monitor platform activity, including total users, monthly job posts, application trends, and top recruiting companies. Built with responsive design principles, the platform ensures accessibility across devices and delivers a secure, scalable, and intelligent job-matching experience for all users.

Keywords: Job Portal, MERN Stack, Resume Analysis, Naive Bayes, User Authentication, Career Insights

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to all those who contributed to the creation and development of this documentation for this website. This endeavor would not have been possible without the collaborative efforts and support of numerous individuals. Though it would not be possible to acknowledge everyone in their names, out here, Intern would like to take the privilege to pay gratitude to each and every individual associated directly or indirectly with this study. Foremost, it was an honor and great privilege to conduct this study.

Our special thanks goes to Mr. Shatrughan Prasad Gupta, Campus Chief and Mr. Basanta Kumar Baral, Assistant Campus Chief of Kathmandu Shiksha Campus for their valuable inputs regarding format and housing of the report. Our heartfelt gratitude to Mr. Yuba Raj Devkota, Coordinator of BCA, and Supervisor of Kathmandu Shiksha Campus for his ever-rewarding presence in all times of need during study and project work. We also want to pay heartfelt gratitude to all the technical and non-technical members for their continuous support, guidance and providing a platform and creating environment for the development of this project. All my colleagues, seniors and juniors need a special mention for their indispensable role, one way or another, in completion of this study and making this work a pleasant experience. Our sincere thanks extend to all the staff of Kathmandu Shiksha Campus who never showed any hesitation to help us in administrative, technical and statistical aspects associated with the study. It goes beyond saying that this project work would have been nothing without patience.

Thank You

Sincerely,

Laxman Rumba

Table of Contents

List of Abbreviations

CSS	Cascading Style Sheets
HTML	Hypertext Transfer Protocol
JS	JavaScript
MERN	MONGODB EXPRESS REACT NODE
OOP	Object-Oriented Programming
UML	Unified Modeling Language

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Introduction

Our project, Job Portal, is designed to build a smart, scalable, and user-friendly platform that bridges the gap between job seekers and recruiters. Whether you're a passionate individual exploring new career opportunities or a company searching for the right talent, our platform offers the perfect space to connect and grow. Our ultimate goal is to provide an intelligent and personalized job search experience tailored to individual skills, interests, and career goals.

In today’s competitive job market, candidates seek more than just job listings—they want guidance, relevance, and opportunities that align with their professional journey. Our platform addresses this need by using a Naive Bayes-based content filtering algorithm to offer custom job recommendations based on user-uploaded resumes, preferences, and skills.

Job seekers can create profiles, upload resumes, and receive real-time job suggestions curated by AI. The platform also offers AI-generated quizzes to evaluate skills and help users identify growth areas. Insights into industry trends and salary expectations add further value, empowering users to make informed decisions.

For recruiters, our system provides tools to post and manage job listings, review applicants, and track application statuses. Companies can also manage visibility of their posts and update their profiles easily. An intuitive and responsive interface ensures smooth usability across all devices.

An admin dashboard adds oversight and control, offering analytics such as total recruiter/job seeker counts, monthly job post tracking, applicant acceptance rates, and top-performing recruiters.

With this AI-powered Job Portal, we aim to revolutionize how people discover careers and how companies discover talent—creating a smarter, faster, and more personalized recruitment experience for all.

1.2 Problem Statement

- Job seekers face difficulty finding relevant job opportunities tailored to their skills and experience.
- Recruiters struggle to efficiently filter and connect with qualified candidates.

1.3 Objectives

- To provide personalized job recommendations using a Naive Bayes algorithm based on resume analysis.
- To streamline the recruitment process with secure authentication, job management tools, and applicant tracking.

1.4 Scope and Limitations

a. Scope

- To develop a job portal system that allows recruiters to post, edit, and manage job listings.
- To integrate a Naive Bayes-based content recommendation algorithm for providing personalized job suggestions based on resume data.

b. Limitations

- The accuracy of job recommendations is highly dependent on the quality and relevance of the resume data uploaded by users.
- The system does not currently support real-time job crawling or integration with external job listing platforms.

1.5 Development Methodology

The Waterfall methodology is adopted for the development of this AI-powered Job Portal Website, following a linear and sequential approach. Each phase is completed thoroughly before progressing to the next, ensuring a disciplined and structured development process. The project begins with the Requirements Analysis phase, where the functional and non-functional needs of job seekers, recruiters, and administrators are collected. This phase defines essential features such as job posting, resume-based recommendations, OTP registration, and AI-generated quizzes.

Next, during the System Design phase, these requirements are translated into detailed architectural blueprints, including database schemas, API routes, and UI mockups. Both frontend and backend components are planned using the MERN stack—MongoDB, Express.js, React, and Node.js—to ensure scalability and responsiveness.

Following the design phase, the Implementation phase involves coding each system module, including user authentication with JWT and OTP, job management for recruiters, resume upload for job seekers, and the Naive Bayes algorithm for job recommendations. APIs and AI components are integrated to support intelligent features.

Once implemented, the Integration and Testing phase begins. This includes unit testing of modules, integration testing of features like job application and resume parsing, and system-level testing to validate functionality against user requirements.

After successful testing, the Deployment phase involves configuring production environments, deploying the application to servers, and ensuring accessibility and performance for end users.

Finally, in the Maintenance phase, the system is monitored for bugs, performance issues, and user feedback. Regular updates and feature enhancements are carried out to keep the platform secure, efficient, and aligned with evolving user needs.

By following the Waterfall model, the development of this job portal ensures clarity, minimizes risks, and results in a reliable and user-focused recruitment platform.

.

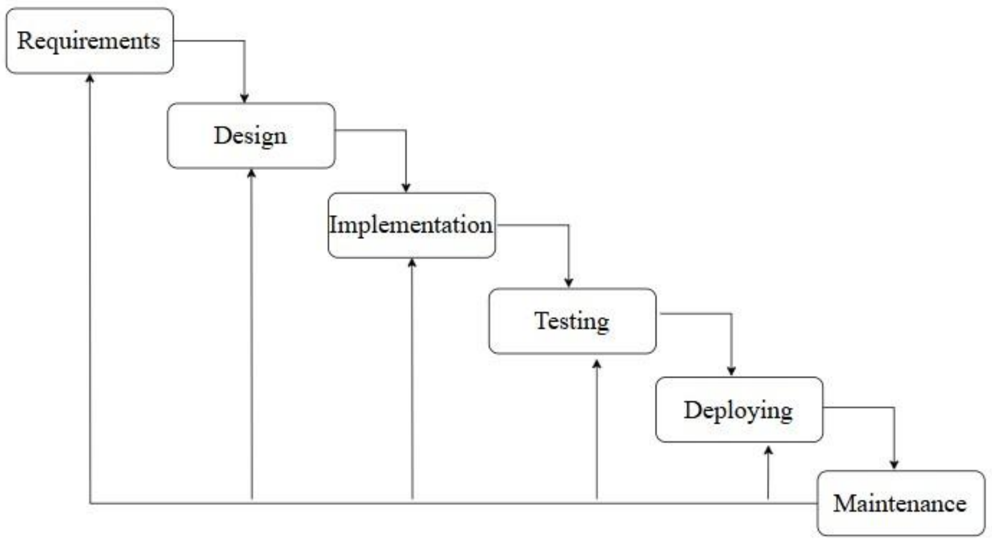


Figure 1.1: Waterfall Methodology

1.6 Report Organization

This project report consists of five chapters altogether. The report has been organized in the order which are described below:

Chapter 1: Introduction

The first part of the report contains the summarized introduction of the whole report. It includes the overall problem statements, objectives, scope and limitation and development methodology of this project.

Chapter 2: Background Study and Literature Review

The second chapter includes background study i.e. description of fundamental theories, general concepts and terminologies related to the project. It also includes the literature review i.e. review of the similar projects, research and theories done by other researchers.

Chapter 3: System Analysis and Design

The third chapter includes the system analysis and design phase in which we should choose between Structured approach or Object-Oriented approach. Then we do requirement analysis, which are functional and non-functional requirements. The functional requirements illustrate using Use-case diagram or Use-case description. It also includes the feasibility analysis about the system which explains whether the system development process is affordable and within the knowledge range of the developers. It shows the technical, operational and economic feasibility of the project development phase. For Object-oriented approach we include Object modeling: Object and Class diagram, Dynamic modeling: State and Sequence diagram and Process modeling: Activity diagram. The explanation of the System designing of the system is also done in this chapter. For this, we use Structured approach or Object-oriented approach. For structured approach we include Architectural design, Database schema design, Interface design (UI/UX) and Physical DFD. For Object-oriented approach we include Refinement of classes and object, Component diagram and Deployment diagram.

Chapter 4: Implementation and Testing

The fourth chapter includes the implementation and testing phase of the proposed system. In the implementation phase, the tools like CASE tools, programming languages and database platforms are implemented. Then, it includes implementation of modules in which we describe the procedures, functions, classes and methods of the project.

Chapter 5: Conclusion and Future recommendation

The fifth chapter includes conclusion and future recommendation. This contains the final paragraphs of the report and in this phase the overall outcome and the developer's point of view is written. The lesson that we learned through all the phases can also be included in this phase.

Chapter 2: Background Study and Literature Review

2.1 Background Study

The online job market has grown significantly, providing job seekers with access to countless opportunities and companies with broader talent pools. However, this rapid expansion has also made the job search process more complex. Job seekers are often overwhelmed by the volume of listings, while recruiters struggle to identify the right candidates efficiently. As a result, the integration of intelligent recommendation systems into job portals has become increasingly important to streamline the recruitment experience [1].

Recommendation systems are designed to analyze user data and provide suggestions that align with user needs and interests. In the context of job portals, they play a crucial role in helping users find relevant job postings based on their background, experience, and preferences. Unlike traditional search methods, which rely on keyword matching, intelligent systems focus on understanding the context and intent of the user to deliver more accurate results [2].

In our project, we have implemented a Naive Bayes classification algorithm to recommend jobs by analyzing resume content uploaded by users. The system examines keywords and textual patterns within resumes, classifies them into predefined job categories, and then suggests relevant job listings accordingly. This probabilistic, supervised learning approach improves accuracy by considering real resume content rather than relying solely on user activity or profile data [3].

This method offers a practical and scalable way to personalize job recommendations, especially for new users with limited interaction history. However, it does have limitations—such as the reliance on clear and complete resume data, and the potential for category misclassification in cases of vague or unconventional resumes. Despite these challenges, such AI-driven solutions enhance the overall job-seeking experience and reduce time spent on irrelevant searches [4][5].

2.2 Literature Review

With the evolution of the job market and the digitization of recruitment processes, job portals have become one of the most widely used platforms for both job seekers and employers. These platforms aim to match candidates with relevant opportunities, but as the volume of users and listings increases, so does the complexity of making the right match.

To enhance this process, researchers and developers have turned to machine learning and AI-based recommendation systems. These systems aim to personalize job search experiences, reduce manual effort, and improve the chances of a successful match.
Thannimalai and Zhang [6] proposed a hybrid recommendation model that combined item-based collaborative filtering and content-based filtering, powered by a Naïve Bayes classifier. Although the context of their study was in tourism, the underlying methodology—classifying and recommending based on a combination of user preferences and data attributes—is directly applicable to job portals. In a similar manner, a job portal can analyze a candidate's resume to classify them into a suitable job category and recommend relevant listings.
Roopesh and Bomatpalli [7] emphasized that recommender systems are among the most valuable applications of machine learning. Their study discussed how these systems filter vast datasets to provide users with relevant suggestions based on their behavior, interests, or historical data. The paper noted that while such systems are widely used in entertainment and travel, their potential in the employment domain remains underutilized. This reinforces the significance of intelligent matching systems like the one implemented in this project, which leverages AI to interpret resumes and recommend jobs.
In an innovative approach, Srisawatsakul and Boontarig [8] developed a recommender system that analyzed users' Instagram photos to understand their preferences without any explicit input. This idea of implicit user profiling is extremely relevant to modern job portals, where many users may not fully complete their profiles or struggle to articulate their ideal job. By using resume data, application behavior, and past interactions, a job portal can similarly build a dynamic user profile and adapt recommendations accordingly.
Rula [9] provided a taxonomy of e-tourism systems that classified how smart technologies manage and utilize user data. Although focused on travel, this study highlighted the broader importance of structured data classification, especially in platforms with large, diverse datasets. Applying this to the job domain, a job portal must manage structured information like job roles, experience levels, locations, and qualifications—all of which must be accurately parsed and understood for personalized recommendations to work effectively.
Banerjee [10] focused on the implementation of a learning model in the service industry, showcasing how AI agents can simulate human decision-making when assisting users. This is particularly relevant in job matching, where human recruiters often assess personality, soft skills, and suitability based on resume language and tone. A well-trained machine

learning model, like the Naïve Bayes classifier used in this project, can begin to simulate such assessments by learning from labeled datasets.

. For example, Zheng [11] used historical tourism data to generate 56 distinct features for recommendation, and through information theory and entropy ranking, was able to reduce complexity while improving accuracy. In the job context, resume parsing may yield dozens of features—skills, certifications, industries, experience levels—so being able to identify and rank the most impactful attributes is key for building a scalable, intelligent system.

Finally, Lops et al. [12] emphasized the effectiveness of content-based filtering in domains where item features (like job descriptions) and user preferences (like resume content) can be clearly mapped. Their research validated the use of machine learning techniques to model user profiles and match them with opportunities that share similar attributes. For job seekers, this approach ensures that recommendations are grounded in their actual strengths and qualifications, rather than just crowd-sourced popularity or clicks

Chapter 3: System Analysis and Design

3.1 System Analysis

Requirement analysis is a vital phase of the software development lifecycle that involves collecting and documenting the expected functionalities, limitations, and constraints of a system. For this job portal project, the requirement analysis focused on the needs of different stakeholders (job seekers, recruiters, and admin) and how to integrate AI-driven recommendations using resume data. This section identifies both functional and non-functional requirements to ensure the system is robust, scalable, and user-friendly.

3.1.1Requirement Analysis

Requirement analysis is a critical phase in software development that involves gathering, analyzing, and documenting the needs and constraints of the project. For the Job Portal recommendation project, the requirement analysis process aimed to identify the key features and functionality of the system, as well as any constraints or limitations that needed to be considered during development.

i. Functional requirements

The functional requirements define the system behavior from a user's perspective and outline the core features of the job portal:

- **User Authentication:**

  The system allows both job seekers and recruiters to register and log in using JWT-based authentication. During the first-time registration, users must verify their account using an OTP sent to their email. Once authenticated, users can securely log in, access their respective dashboards, and log out when finished. Invalid credentials or unverified accounts will result in access denial, accompanied by appropriate error messages such as “Invalid OTP” or “Incorrect email/password.

- **Resume Upload and Profile Management:**

  Job seekers can upload their resumes in PDF or DOC format. The system processes this data to extract relevant information such as skills, experience, and preferred roles. Based on the parsed data, users can edit and update their personal profiles to improve recommendation accuracy.

• **Job Recommendations:**

The system uses a Naïve Bayes classification algorithm to categorize resumes and suggest relevant job listings. This personalized recommendation engine allows job seekers to explore positions that align with their skills, interests, and qualifications, enhancing the job-finding experience.

• **Job Search and Applications:**

Users can browse and search available jobs based on filters like job title, location, company, and experience level. Once they find a suitable position, they can apply directly through the portal. Users can also track the status of their applications (e.g., pending, accepted, or rejected) from their dashboard.

• **Recruiter Dashboard and Job Posting Management:**

Recruiters can log in to their dashboard to post new jobs, edit existing listings, and view all their active or expired posts. They can mark job posts as visible or hidden. Additionally, recruiters can view and manage the list of applicants for each job, and take actions such as accepting or rejecting candidates.

• **Admin Dashboard and Monitoring:**

Admins can access an overview dashboard where they can view key platform statistics, such as the total number of job seekers, recruiters, monthly job posts, and application outcomes. The system also identifies top-performing companies and provides visual insights into application trends (accepted, pending, rejected).

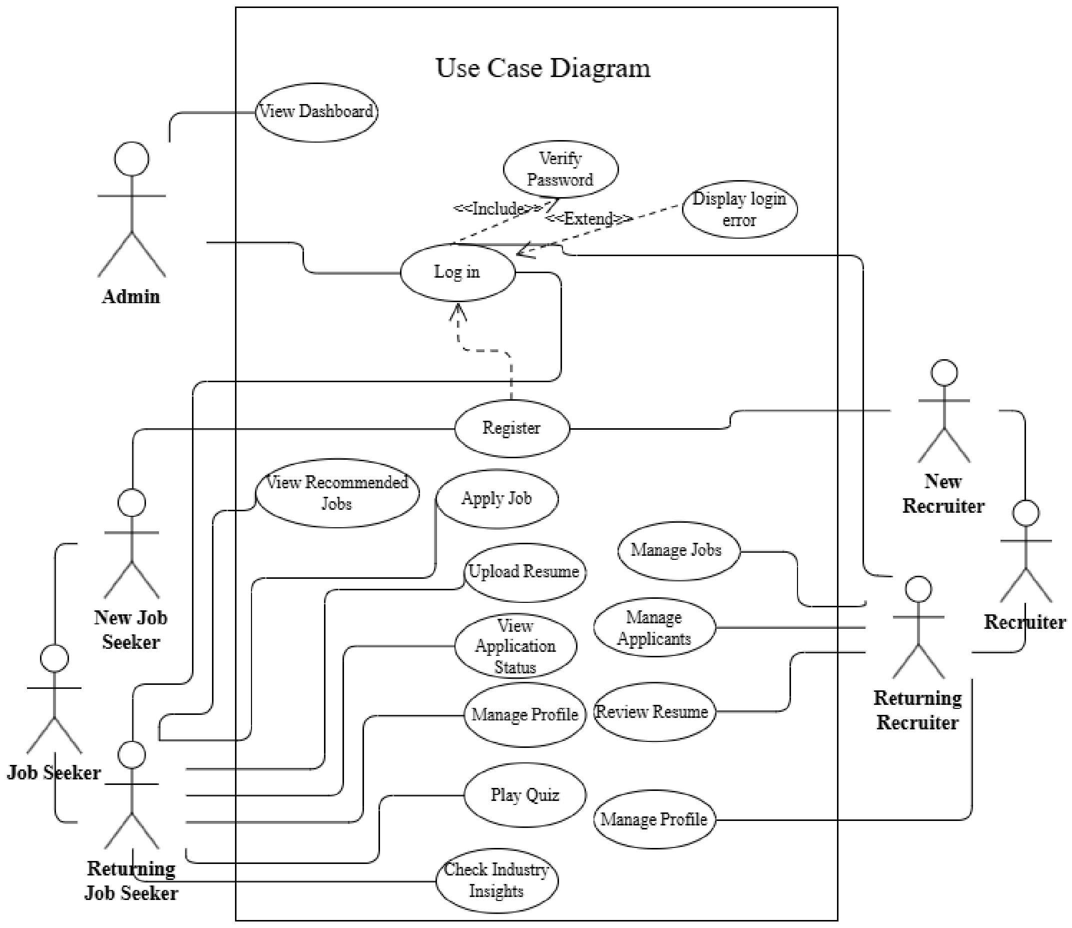


Figure 3.1: Use-case Diagram

ii. Non-functional requirements

Non-functional requirements define the quality attributes and performance standards that the system must meet. These are crucial to ensure the system is not only functionally correct but also user-friendly, secure, scalable, and maintainable over time. The non-functional requirements for the job portal are as follows:

- Usability:

  The system is designed with a clean, intuitive, and user-friendly interface to ensure ease of navigation for all user roles—job seekers, recruiters, and administrators. Pages are logically structured with consistent layouts and accessible action buttons, allowing users to perform tasks such as registration, login, job search, and application with minimal effort.
- Performance:

  The platform ensures fast response times, optimized server requests, and minimal latency even during peak usage. Efficient database queries and lightweight frontend assets contribute to quick page loads and real-time interactions. The system is capable of handling multiple concurrent users without performance degradation.

- Accessibility:

  The website is accessible to users with disabilities, complying with web accessibility standards and guidelines, and offering assistive technologies such as screen readers or keyboard navigation.

- Compatibility:

  The website is cross-browser and cross-device compatible, functioning seamlessly across modern web browsers (Chrome, Firefox, Safari, Edge) and devices (desktops, tablets, and smartphones). Responsive design principles are employed to ensure an optimal viewing experience on all screen sizes.

- Maintainability:

  The system follows industry-standard coding practices with modular, well-commented code and clear documentation. Version control (e.g., Git), automated testing, and continuous integration/deployment pipelines are implemented to streamline the maintenance, debugging, and feature expansion processes.

- Security:

  Robust security measures are implemented to protect sensitive user information. These include JWT-based authentication, OTP verification during registration, HTTPS communication, hashed passwords, role-based access control, and secure file uploads.

- Reliability:

  The platform is designed for high availability and fault tolerance, ensuring that users can rely on consistent system performance. Proper error handling, backup mechanisms, and monitoring tools are in place to detect and resolve issues proactively.

3.1.2 Feasibility Analysis

a) Technical Feasibility

The system is technically feasible as it leverages widely available open-source tools and technologies, primarily the MERN stack (MongoDB, Express.js, React, and Node.js), which are well-supported and documented. These technologies are sufficient to implement all planned functionalities, including JWT-based authentication, AI-driven resume classification, and real-time job matching.

The development environment requires only minimal system resources, and no proprietary or high-cost platforms are necessary. The technical specifications for development and deployment are as follows:

- Operating System: Windows, macOS, or any Linux-based OS
- RAM: 4GB or higher
- Disk Space: 1GB or higher
- Devices: Desktop, Laptop, or Android Smartphone
- Browser Compatibility: Chrome, Firefox, Safari, Edge
- Internet Requirement: Minimum 1 Mbps connection for smooth user experience

Given the accessibility of these tools and the development team's familiarity with JavaScript-based technologies, the system poses no significant technical challenges.

b) Operational Feasibility

Operational feasibility assesses how well the proposed system integrates into day-to-day operations and how efficiently it can be used by its intended stakeholders. The job portal is designed with a user-friendly and intuitive interface for job seekers, recruiters, and administrators, ensuring ease of navigation and operation.

The workflow—from user registration and job posting to job recommendations and candidate management—is streamlined and tested for practical use. The system can be used by organizations of various sizes without the need for extensive training or technical knowledge. Furthermore, administrative features such as dashboards, analytics, and applicant tracking ensure that the platform can support real-world recruitment operations effectively.

Given the clarity of the system's purpose and the accessibility of its interface, the platform is operationally feasible for regular use

.

c) Economic Feasibility

The system is highly economical as it utilizes open-source software frameworks and libraries, avoiding licensing or subscription fees. Development and deployment can be done on low-cost cloud hosting platforms or even local servers for small-scale usage. There is no need for specialized hardware, and ongoing maintenance costs are minimal due to the use of modular, maintainable code and community-supported tools.

In addition, the system reduces the manual workload of recruiters and enhances job seeker satisfaction through personalized recommendations, leading to potential long-term cost savings for organizations using the platform. The low upfront cost combined with high potential value makes the job portal project economically feasible

.

3.1.3 Object Modeling: Object and Class Diagram

The class diagram for this Job Portal System represents the key entities and their relationships within the platform. At the core, the User class encapsulates personal details, skills, education, and experience, while also maintaining references to assessments taken and industries of interest. The Company class models employers, storing information such as name, email, password, and a list of associated job postings represented by the Job class, which includes attributes like title, description, salary, location, category, and deadline. Users apply to jobs via the Application class, which forms associations between users, companies, and jobs while tracking application status. The Admin class enables administrative access and control, including user and job management. Additionally, the system features an Assessment class to store quiz scores and improvement tips for users, and an IndustryInsight class powered by AI to provide data like salary trends and in-demand skills for various industries. The OTP class supports email-based verification during signup, while JobRoleCategory helps classify job postings. Together, these interconnected classes illustrate how users, recruiters, and administrators interact within the system to manage careers, hiring, and insights effectively.

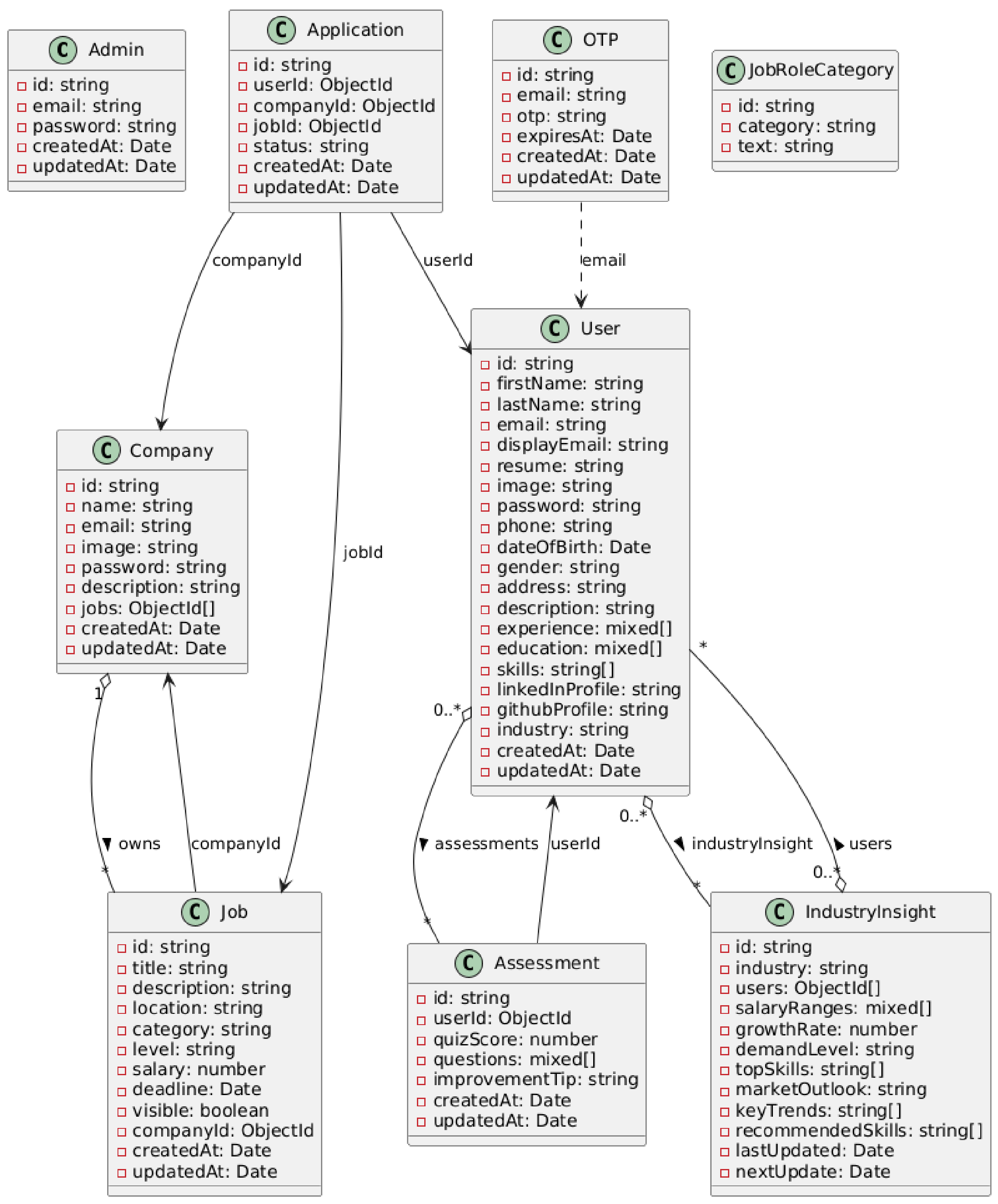


Figure 3.2: Class Diagram of JobPortal

3.1.4 Dynamic Modelling: State and Sequence Diagram

The state diagram represents the flow of user experience within our Job Portal System. It begins at the Idle state, where users can either log in or register depending on their status. Upon successful registration, users undergo OTP verification to confirm their identity. After login or verification, users are directed to their respective dashboards—Job Seeker, Recruiter (Company), or Admin—based on their roles. Job Seekers can browse job listings, apply for positions, track application status, take AI-generated quizzes, or explore industry insights. Recruiters can post jobs, manage listings, and review or update applicant statuses. Admins have access to platform-wide analytics, including user statistics, job activity, and top recruiter insights. All users have the ability to edit their profiles and securely log out, returning them to the Idle state. This structured flow ensures personalized, secure, and efficient navigation throughout the platform for all user types.

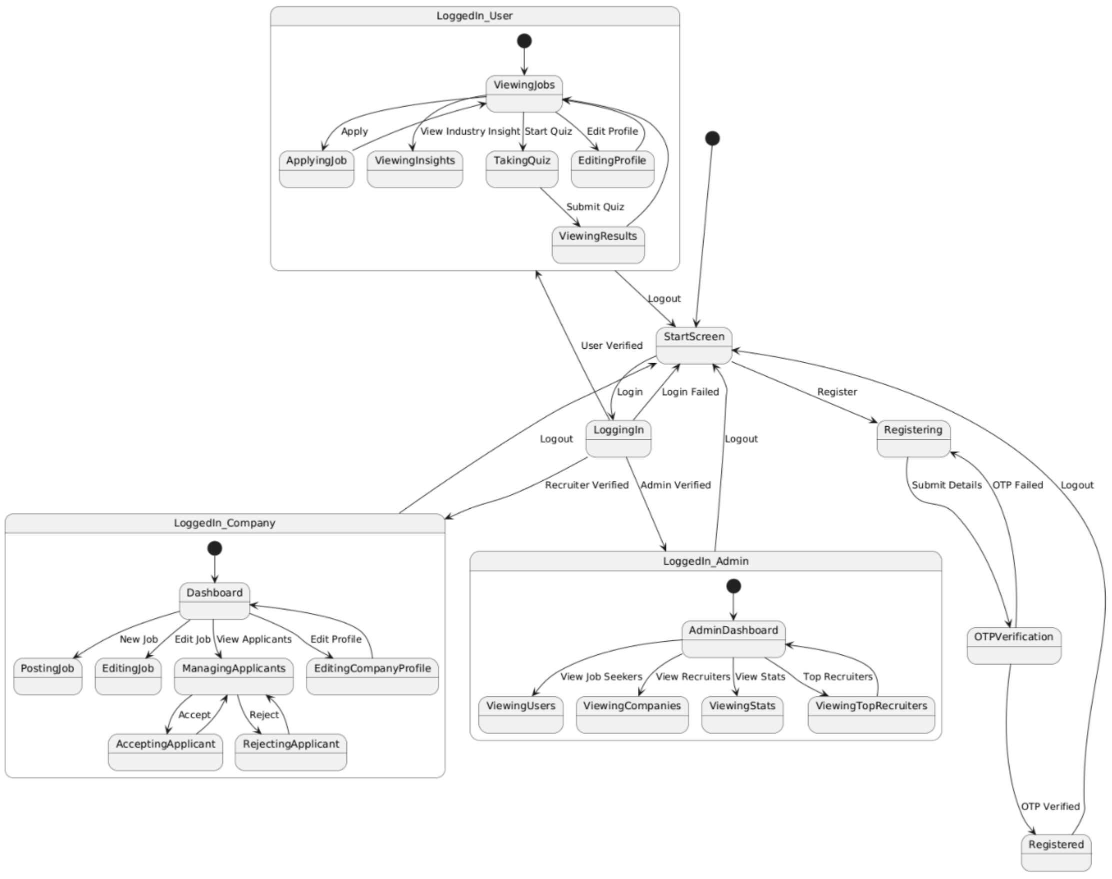


Figure 3.3: State Diagram of Job Portal

Sequence diagrams for the Job Portal Recommendation System are a type of Dynamic Modeling tool that visually capture the message flow between objects in the system over time. The sequence diagram provides a visual representation of the dynamic behavior of the system during a specific user interaction, helping to understand how various components communicate.

In our platform, the interaction begins when a user (job seeker) logs in. They send a login request to the system, which checks the credentials and responds with confirmation if valid. Once logged in, the user can browse available job listings. The system retrieves these listings from the job service and displays them to the user. If a user finds a job they're interested in, they can view more details, prompting the system to fetch and show additional job information.

When the user decides to apply for a job, they initiate an application request. The system records this application as “Pending” in the database. At this point, the recruiter (company) receives the application through their dashboard. The recruiter can review the applicant’s details and either accept or reject the application. This decision is updated in the system and reflected in the job seeker’s application status.

Meanwhile, the admin may monitor platform activity, including user statistics, application trends, and top recruiters. Throughout the process, the system ensures that each interaction — from login and job search to application and response — is seamless, well-coordinated, and provides real-time feedback to all users involved

.

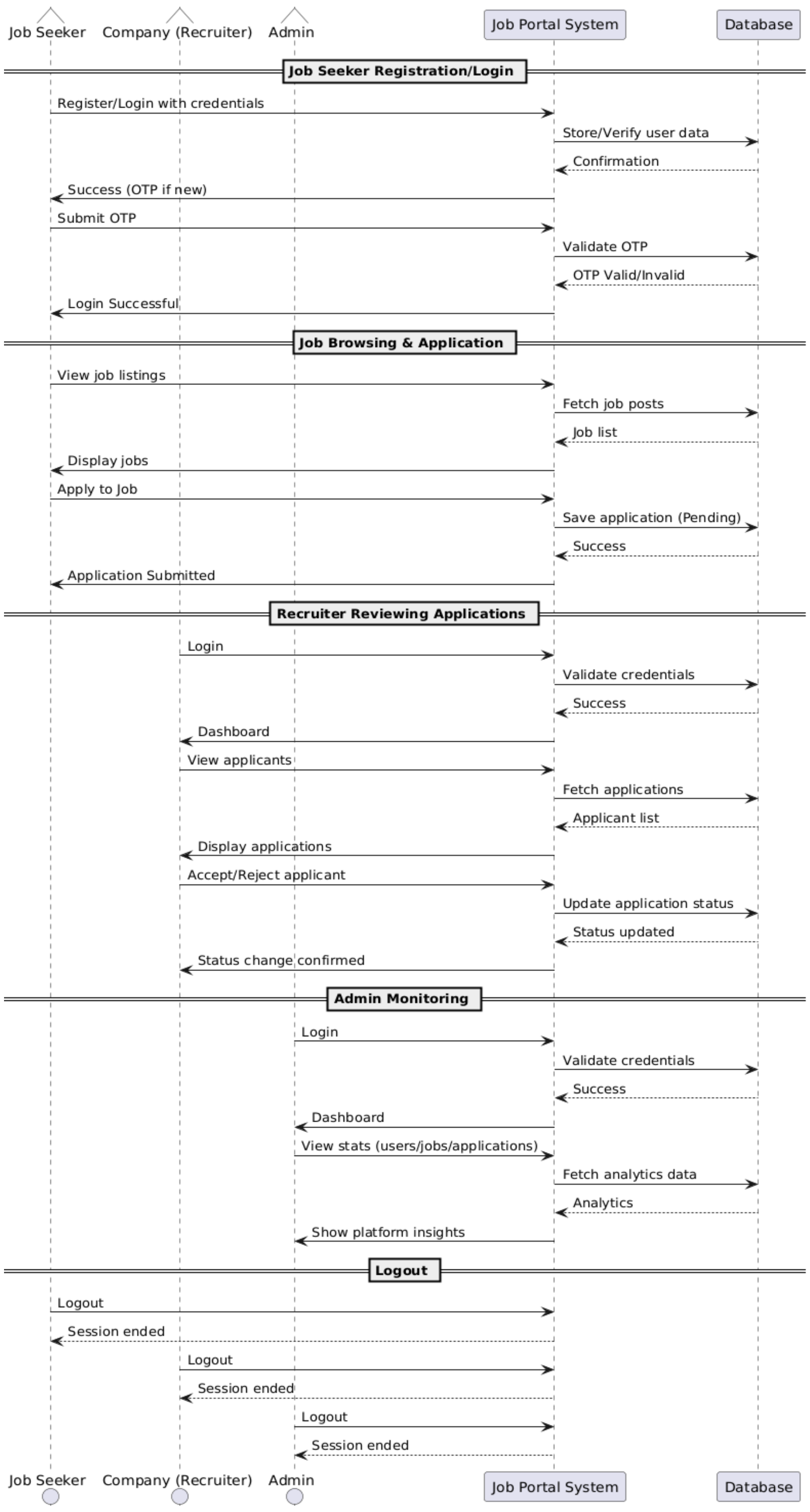


Figure 3.4: Sequence Diagram of Job Portal

3.1.5 Process modeling: Activity Diagram

Activity diagrams act like blueprints for processes, depicting the flow of activities and decisions from start to finish. An activity diagram in UML is a type of diagram that visually presents a series of actions or flow of control in a system. It shows workflows of stepwise activities with support for choice, iteration, and concurrency.

This activity diagram illustrates the complete flow of operations in a Job Portal Recommendation System, highlighting the interactions for job seekers, recruiters (companies), and admins. The process begins with user registration, where job seekers and recruiters sign up by entering their personal or company details. Upon successful registration, users log in with their credentials, which the system verifies.

If login credentials are invalid, the system prompts the user to re-enter them. Once authenticated, users are directed to their respective dashboards based on their role—admin, job seeker, or recruiter.

Admins can access platform statistics, monitor application statuses, and view top recruiters. Recruiters (companies) can post jobs, edit listings, view applications, and accept or reject candidates. Job seekers, on the other hand, can manage their profiles, upload resumes, explore job listings, apply for jobs, and view application history.

The system also offers AI-powered features for job seekers, including personalized job recommendations, industry insights, and skill-based quizzes. Each role has access to a secure logout function, ensuring safe session termination. This diagram effectively captures the user journey and operational flow, ensuring clarity and well-structured interaction across all user roles in the system.

.

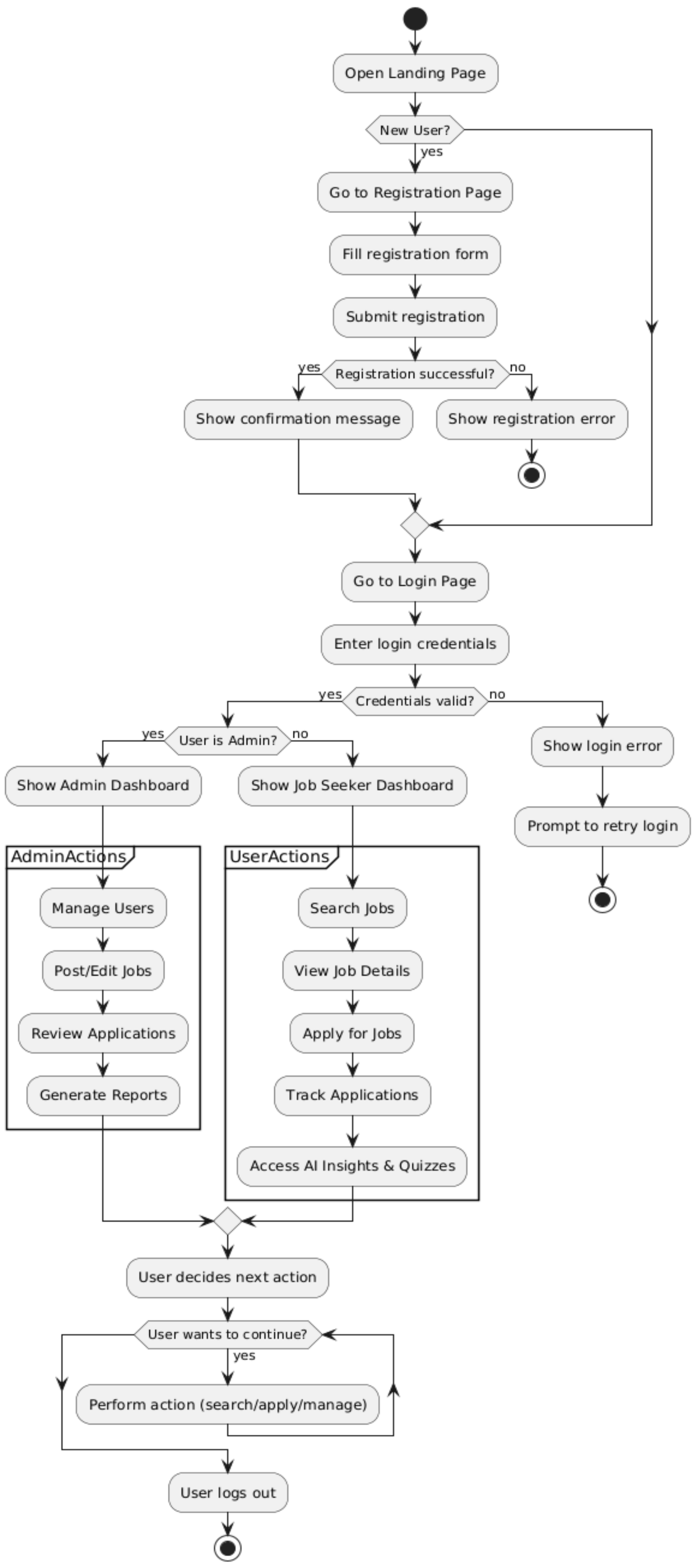


Figure 3.5: Activity Diagram for Job Portal

3.2 System Design

System design is the process of defining the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system. One of the main components of software design is the software requirement analysis.

3.2.1 Refinement of Classes and Object

One of the main components of software design is the software requirement analysis. In this project there is one database used for store the information of users who have register their account in system.

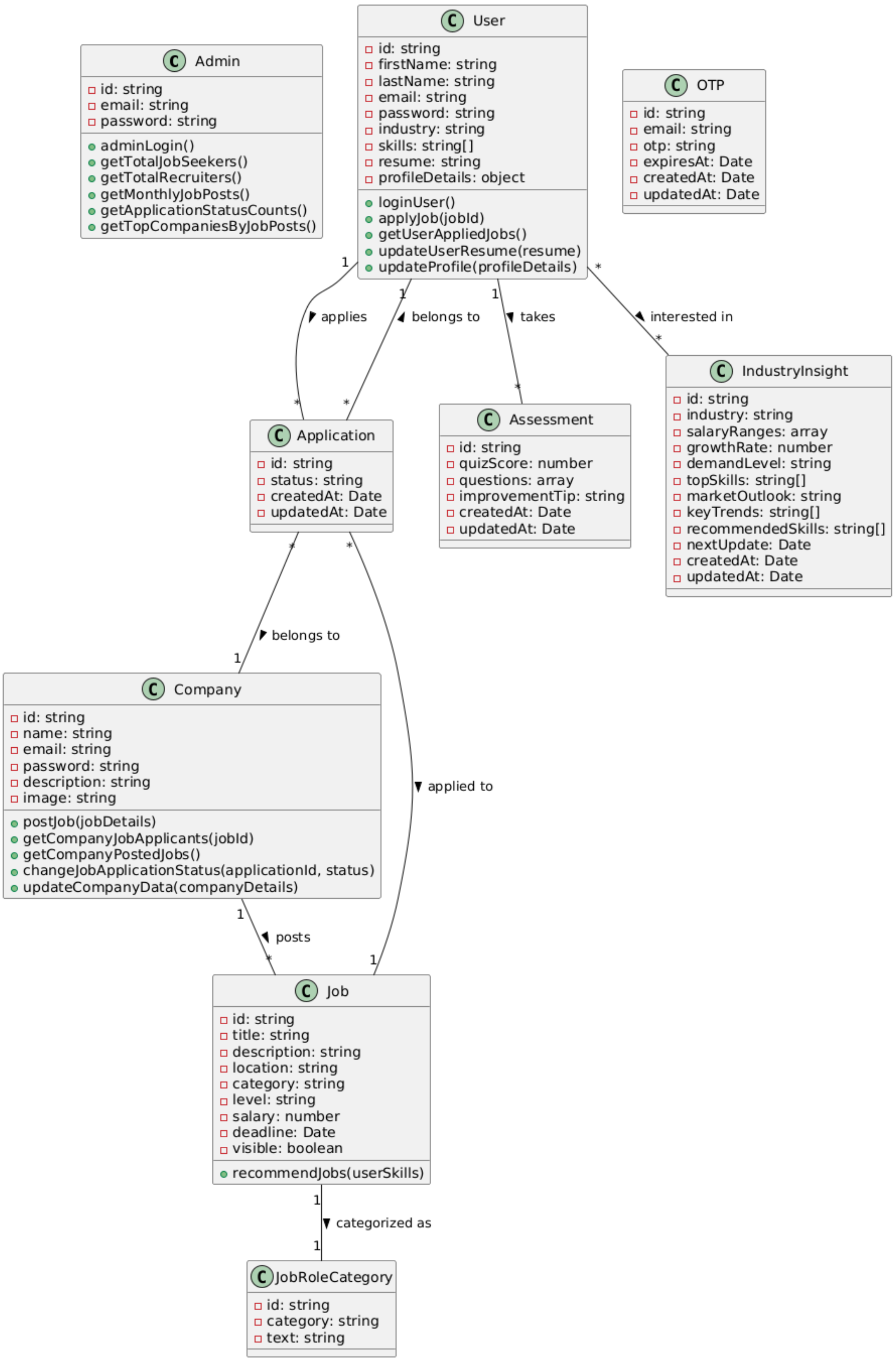


Figure 3.6: Refined Class Diagram

3.2.2 Component Diagram

A component diagram for fake news detection is used to break down a large object-oriented system into the smaller components, so as to make them more manageable. A component diagram is a type of UML diagram that shows the structural relationships and dependencies between the components of a software system. It illustrates how software components are connected and interact with each other within a system. Components can represent individual modules, libraries, executables, or other parts of a system.

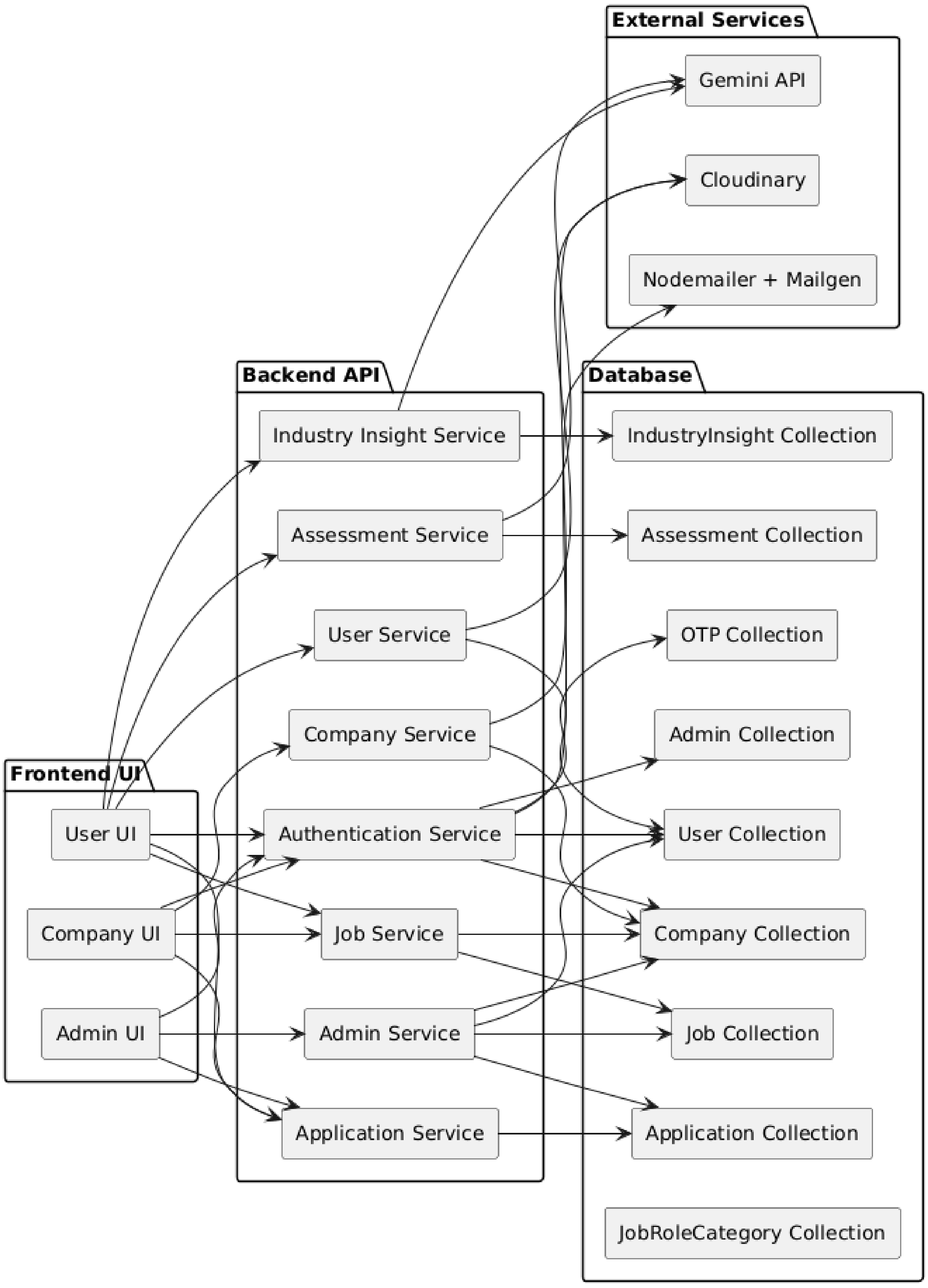


Figure 3.7: Component Design

From Figure 3.7, the component diagram illustrates the architecture of the job portal system, showing how different components interact across the frontend, backend, and supporting services. The Frontend UI is divided into three main interfaces: Admin UI, Company UI, and User UI—each built with modern web technologies like React. These interfaces handle user interactions, form submissions, and dynamic content rendering.

The frontend communicates with the Backend API, which consists of specialized services such as Authentication, User, Admin, Company, Job, Application, Assessment, and Industry Insight services. Each service is responsible for a specific set of operations, ensuring a clean separation of concerns. For instance, the Authentication Service manages secure login and OTP verification using JWT and Nodemailer with Mailgen, while the Assessment and Insight services integrate with the Gemini API to generate personalized quizzes and industry insights.

The backend also interacts with the MongoDB Database, where various collections like User, Company, Job, Application, and others are maintained. These collections store persistent data which is queried and updated by respective services. Additionally, media uploads (like resumes and profile pictures) are managed through Cloudinary, ensuring efficient and scalable file handling.

This component diagram not only outlines the responsibilities of each system module but also visualizes the seamless flow of data between user interfaces, backend services, third-party APIs, and the database. It highlights how modular architecture ensures maintainability, security, and scalability within the job portal system

.

3.2.3 Deployment Diagram

The deployment diagram of the Job Portal system portrays the static deployment view of a system. It involves the nodes and their relationships. Deployment diagram are UML structural diagrams that shows the relationships between the hardware and software components in the system and the physical distribution of the processing i.e., Deployment diagram are used to visualize the topology of the physical components of the system where software components are deployed.

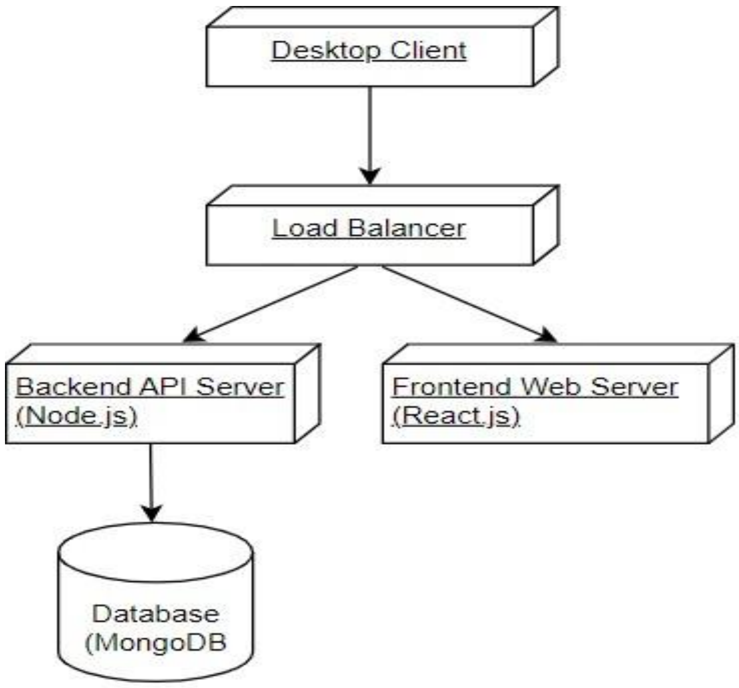


Figure 3.8: Deployment Diagram of Job Portal

From Figure 3.8, the Deployment Diagram of the Job Portal System illustrates how various components are deployed and interact within a scalable cloud-based environment. The system is hosted on a modern cloud platform, ensuring high availability and performance for end users. At the entry point, a Load Balancer evenly distributes incoming traffic across multiple Backend API Servers, enhancing the scalability and fault tolerance of the platform.

The Backend API Server, developed with Node.js, is responsible for handling core business logic, including user management, job postings, applications, AI-driven assessments, and industry insights. These servers also interact with external services such as the Gemini API for content generation, Cloudinary for media file storage, and Nodemailer with Mailgen for OTP and email communication.

Simultaneously, the Frontend Web Server, built with React.js, delivers the user interface to job seekers, recruiters, and admins. It ensures a smooth and responsive user experience by managing routing, rendering, and client-side interactions.

The MongoDB Database acts as the persistent data store, housing collections such as Users, Jobs, Applications, Companies, Assessments, and Industry Insights. Its NoSQL structure supports flexible and efficient data querying for a dynamic and rapidly growing user base.

This deployment architecture ensures the Job Portal System remains robust, scalable, and secure—capable of handling increasing loads while delivering real-time responses and seamless user experiences.

3.3 Algorithm details

The project implements a Naive Bayes classification algorithm to recommend relevant job categories to users based on the content of their resumes. Naive Bayes is a probabilistic machine learning algorithm based on Bayes' Theorem, widely used for text classification due to its simplicity and effectiveness.

This algorithm processes textual data extracted from user resumes, applying preprocessing steps such as lowercasing, stopword removal, and synonym normalization. After preprocessing, it classifies the resume content into predefined job role categories by calculating the probabilities of each category given the text features.

3.3.1 Text Preprocessing

Before classification, the text undergoes several preprocessing steps:

- **Lowercasing:** All text is converted to lowercase to ensure uniformity.
- **Stopword Removal:** Common words that do not contribute meaningful information (e.g., "and", "the", "of") are removed.
- **Synonym Normalization:** Different terms with similar meanings (e.g., "ReactJS" and "React") are normalized to a common representative word to improve consistency

3.3.2 Training the Naive Bayes Model

The model is trained using a dataset of job role categories with representative textual descriptions. For each category:

- The frequency of each word occurring in its descriptions is counted.
- The prior probability of the category is calculated based on its occurrence frequency.
- Word likelihoods (probability of each word given the category) are estimated.

These statistics form the basis for classification and are updated during training to improve prediction accuracy.

3.3.3 Classification and Scoring

When a user's resume text is input to the system:

- The same preprocessing steps are applied.
- The algorithm calculates the logarithmic probabilities of the text belonging to each category using Bayes' theorem, summing the prior and conditional word probabilities.
- Categories are scored and ranked based on these computed probabilities.
- The top categories represent the most probable job roles suited to the user's skills and experience.

3.3.4 Job Recommendation

Using the predicted job categories from the classification step, the system:

- Retrieves active job listings tagged with matching categories.
- Scores and ranks jobs based on their relevance to the user's classified preferences.
- Provides the user with a personalized list of recommended jobs most aligned with their resume content.

```
async function trainNaiveBayes() {
  const JobRoleCategory = require("../models/JobRoleCategory");
  const trainingData = await JobRoleCategory.find();
  categoryWordCounts = {};
  categoryCounts = {};
  totalWords = 0;

  trainingData.forEach(({ text, category }) => {
    const words = preprocess(text);

    if (!categoryWordCounts[category]) {
      categoryWordCounts[category] = {};
      categoryCounts[category] = 0;
    }

    categoryCounts[category] += 1;
    words.forEach((word) => {
      categoryWordCounts[category][word] =
        (categoryWordCounts[category][word] || 0) + 1;
      totalWords += 1;
    });
  });
  console.log("Training complete!");
}

function classifyJob(text) {
  const words = text.toLowerCase().split(/\s+/);
  const scores = {};

  Object.keys(categoryCounts).forEach((category) => {
    let probability = Math.log(categoryCounts[category] / totalWords);

    words.forEach((word) => {
      const wordCount = categoryWordCounts[category][word] || 0;
       const wordProbability = Math.log((wordCount + 1) / (totalWords +
1));
      probability += wordProbability;
    });

    scores[category] = probability;
  });

  return scores;
}
module.exports = { trainNaiveBayes, classifyJob };
```

This approach ensures personalized, accurate job recommendations by leveraging the probabilistic Naive Bayes classifier to understand user profiles from resume text, making the recommendation process both efficient and interpretable.

Key Components:

1. **User Resume Profile:**

The user resume profile is a textual representation of the user's skills, experience, and qualifications extracted from their uploaded resume. The text undergoes preprocessing steps including lowercasing, stopword removal, and synonym normalization to standardize the content. This processed text serves as the input for classification into job role categories.

2. **Job Role Categories:**

Predefined job role categories represent different classes such as "Software Developer," "Data Analyst," "Project Manager," etc. Each category has associated representative text samples used to train the Naive Bayes classifier. The training process involves calculating the frequency of words within each category, forming a probabilistic model of word-category relationships

3. **Naive Bayes Classification Formula:**

The probability of a category CCC given a resume text TTT is calculated using Bayes' theorem and the naive assumption of conditional independence of words:

$$P(C|T) \propto P(C) \times \prod_{i=1}^{n} P(w_i|C)$$

Where:

- P(C) is the prior probability of category C.
- P(wi|C) is the likelihood of word wi given category C.
- n is the number of words in the resume text.

In practice, logarithms are used to avoid numerical underflow:

$$\log P(C|T) = \log P(C) + \sum_{i=1}^{n} \log P(w_i|C)$$

4. **Job Recommendation Generation:**

- o After classifying the user resume, the top categories with the highest probabilities are identified.
- o Active job listings tagged with these top categories are fetched.
- o Jobs are ranked by their relevance scores corresponding to the predicted categories.
- o The system recommends the highest-ranked jobs to the user, aligning with their skills and experience

.

5. **Implementation Example:**

Assume a user's resume contains keywords related to "JavaScript," "React," and "Node.js." After preprocessing, the classifier calculates probabilities:

- *P(Software Developer|Resume)=−12.3*
- *P(Data Analyst|Resume)=−25.*
- *P(Project Manager|Resume)=−40.*

Since the Software Developer category has the highest log-probability, jobs tagged as Software Developer are retrieved.

For each job:

- The job description text is preprocessed and classified similarly.
- Jobs with categories matching the user's top categories are scored.
- The system ranks jobs by these scores and recommends the best matches.

This Naive Bayes classification algorithm effectively personalizes job recommendations by analyzing the content of user resumes and matching them to suitable job categories. By leveraging probabilistic text classification, it provides accurate and relevant job suggestions that improve the user's job search experience

Chapter 4: Implementation and Testing

4.1 Implementation

4.1.1 Tools used

1. Design and Modelling Tools

- **Draw.io**

  A versatile, web-based diagramming tool, draw.io in our project is used for creating various types diagrams including use-case diagram, class diagram, sequence diagram and activity diagram. Draw.io offers an intuitive interface and extensive libraries of shapes and connectors for effective modeling and design.

2. Development and Integration Tools

- **Visual Studio Code**

  Using Visual Studio Code (VS Code) as the primary development environment for our Job Portal project greatly enhances the development workflow. VS Code offers powerful features such as integrated debugging, Git version control, extensions, and a customizable interface, making it an ideal tool for building and maintaining our application. With VS Code, we can efficiently write, test, and debug both frontend and backend code, ensuring a smooth and cohesive development experience throughout the project lifecycle.

3. Programming Languages

- **HTML**

  HTML (Hypertext Markup Language) provides the foundational structure for all web pages in our job portal. It defines elements like headers, forms, buttons, and links, ensuring the content is well-organized and accessible. By using semantic HTML, the website is easy to navigate and optimized for search engines. This foundation allows CSS and JavaScript to enhance the interface and interactive features, delivering a seamless user experience from job search to application submission.

- **Tailwind CSS**

  Tailwind CSS is a utility-first CSS framework used to create a clean, modern, and highly customizable user interface in our job portal. Instead of writing extensive custom CSS, Tailwind allows us to apply predefined utility classes directly to HTML elements.

This accelerates the design process, ensures consistent styling across pages, and enables rapid prototyping with responsive and mobile-friendly layouts. Tailwind CSS helps us deliver an attractive and user-friendly design that adapts to various devices.

- **JavaScript**

  JavaScript powers the interactive and dynamic functionality of our job portal on both frontend and backend. On the frontend, JavaScript with React.js manages the user interface, state, and user interactions such as job searches, profile updates, and application tracking without full page reloads. On the backend, JavaScript runs on Node.js, handling API endpoints, processing job applications, managing authentication, and interacting with the MongoDB database. Using JavaScript across the stack ensures consistency, faster development, and a responsive user experience.

- **React.js**

  React.js is used to build a dynamic and responsive user interface for the job portal. We develop reusable components such as job listings, search bars, application forms, and user dashboards. This modular architecture makes the codebase organized, maintainable, and scalable. React's virtual DOM efficiently updates components for smooth interactions, while hooks like useState and useEffect manage state and side effects seamlessly. React enables a fast, interactive, and user-centric experience critical to engaging job seekers and recruiters.

- **Express.js**

  Express.js serves as the backend web framework for our job portal, facilitating the creation of RESTful APIs and server-side logic. It manages routing for operations such as user authentication, job posting, application submission, and profile management. Express acts as middleware between the React frontend and the MongoDB database, processing HTTP requests and responses efficiently. It also supports middleware for authentication, validation, error handling, and security features. By leveraging Express.js, we ensure a scalable and robust backend that supports the portal's dynamic functionalities.

- **Node.js**

  Node.js is used to enabling us to execute server-side JavaScript code, in our project. It handles the backend logic of our application, allowing us to create a robust and scalable server environment. Node.js processes incoming requests from the frontend, manages data operations with MongoDB, and ensures efficient performance through its non-blocking I/O model. This means that even when handling multiple concurrent requests, Node.js can process them asynchronously without slowing down the system. By using Node.js, we ensure that our server can handle user authentication, booking management, and other essential functions smoothly and efficiently, providing a seamless experience for our users.

4. Database Tools

- **Mongo DB**

MongoDB, a NoSQL database, is used in our project to manage and store all critical data, including user profiles, job postings, company details, and job applications. Unlike traditional relational databases, MongoDB uses a flexible, document-oriented model to store data in JSON-like documents. This flexibility is especially advantageous for our job portal, as job listings and user profiles often contain varied and evolving sets of attributes. MongoDB's schema-less nature allows us to accommodate these differences without rigid schema constraints, making it easier to adapt and scale the application as new features and data types emerge.

To interact with MongoDB, we use the Mongoose library within our Node.js backend environment. Mongoose provides a clear and efficient way to define schemas and models, enforcing consistent data structures while still benefiting from MongoDB's flexibility. This integration enables us to perform complex queries, validations, and transactional operations smoothly.

When job seekers search for jobs, apply for positions, or update their profiles, and when recruiters post or manage jobs, the backend leverages Mongoose to efficiently access and manipulate the relevant data. This results in a responsive and seamless user experience.

By harnessing MongoDB's robust data management capabilities, our job portal can easily scale and evolve to meet the needs of both job seekers and recruiters, supporting a dynamic and feature-rich platform.

4.1.2 Implementation details of modules

1. Registration Module

The registration module in our job portal system is implemented using Node.js and Express.js to provide secure user authentication and authorization for job seekers, companies, and admins. Users register by submitting their details such as name, email, and password, which is securely hashed before storing in the database. During registration, a one-time OTP is sent to the user’s email for verification to ensure authenticity. For login, users provide their email and password, which are verified against stored credentials, and upon success, a JWT token is issued to maintain secure sessions. The module also includes middleware to protect routes and validate user inputs, ensuring a safe and reliable authentication process across the platform.

```
exports.registerUser = async (req, res) => {
  const { firstName, lastName, email, password } = req.body;


  const imageFile = req.file;
  if (!firstName || !lastName || !email || !password || !imageFile) {
    return res
      .status(404)
      .json({ success: false, message: "fields are missing" });
  }


  try {
    const existingUser = await User.findOne({ email: email });
    const existingCompany = await Company.findOne({ email: email });
    if (existingUser || existingCompany) {
      return res
        .status(409)
        .json({ success: false, message: "Email already exists" });
```

```
}
const salt = await bcrypt.genSalt(10);
const hashedPassword = await bcrypt.hash(password, salt);

const imageUpload = await cloudinary.uploader.upload(imageFile.path);

const newUser = await User.create({
  firstName,
  lastName,
  email,
  password: hashedPassword,
  image: imageUpload.secure_url,
});

const payload = {
  id: newUser._id,
  email: newUser.email,
  firstName: newUser.firstName,
  lastName: newUser.lastName,
  image: newUser.image,
};

res.status(201).json({
  success: true,
  message: "User registered successfully!",
  company: {
    firstName: newUser.firstName,
      lastName: newUser.lastName,
      email: newUser.email,
      image: newUser.image,
    },
    token: generateToken(payload),
  });
 } catch (error) {
  console.error("Error checking existing company:", error);
  res.status(500).json({ success: false, message: "Internal Server Error" });
 }
};
```

2. Login Module

The login module in our job portal system enables registered users—including job seekers, recruiters, and admins—to securely access their accounts by entering their email and password. This module handles the authentication process by verifying user credentials against the stored, hashed data in the database. Upon successful verification, users are issued a JWT token to maintain their session and are redirected to their respective dashboards. Additionally, the logout functionality ensures users can securely end their sessions, protecting account access and maintaining system security.

```
exports.loginUser = async (req, res) => {
 const { email, password } = req.body;
 try {
  const user = await User.findOne({ email });
  if (!user) {
   return res
    .status(401)
    .json({ success: false, message: "User not found" });
  }
```

```
const isPasswordValid = await bcrypt.compare(password, user.password);
const payload = {
  id: user._id,
  email: user.email,
  firstName: user.firstName,
  lastName: user.lastName,
  image: user.image,
};

if (isPasswordValid) {
  const token = generateToken(payload);
  res.status(200).json({
    success: true,
    message: "login succesful",
    user: {
      firstName: user.firstName,
      lastName: user.lastName,
      email: user.email,
      image: user.image,
    },
    token,
  });
} else {
  return res
    .status(401)
    .json({ success: false, message: "Invalid credentials" });
}
```

```
  } catch (error) {
    console.error("Login error:", error);
    res.status(500).json({ success: false, message: "Internal server error" });
  }
};
```

4. Job Module

The Job module in our Node.js and Express.js application handles essential operations related to job listings:

- Fetch All Jobs: Retrieves a list of all active job postings from the database.
- Post Job: Recruiters can add new job openings by providing details such as title, description, location, salary, and deadline. These job posts are securely saved in the database.
- Update Job: Recruiters can modify existing job listings by specifying updates like changing the job description, salary, or visibility status.
- Delete Job: Recruiters can remove job postings by specifying the job ID, ensuring outdated or filled positions are no longer visible.
- Search Jobs: Job seekers can search for relevant jobs using keywords in job titles or descriptions to quickly find matching opportunities.

Overall, this module efficiently manages all job-related actions, enabling recruiters to maintain job posts and job seekers to find and apply for suitable positions with ease.

4.2 Testing

4.2.1 Test cases for Unit Testing

Following are the tables representing test cases of unit testing:

Table 4.1: Test Case for Registration

ID	Test Case Description	Test Data	Expected result	Actual Result	Test Result
R_1	User enters valid signup information	First Name: Laxman Last Name: Rumba Email: rumba@gmail.com Password: Test@123	Redirect to login page	Redirects to Login Page	Pass

R_2	User enters existing username	First Name: Laxman Last Name: Rumba Email: rumba@gmail.com Password: Test@123	Display Error Message	“User already exists.	Pass

In the above test cases, the system’s registration functionality was evaluated through multiple scenarios (R_1 and R_2). In test case R_1, a user successfully registered by providing valid information, resulting in a redirection to the login page as expected. Test case R_2 verified that when a user attempts to register with an email already in use, the system correctly identifies the duplication and displays the appropriate error message, “User already exists.” Both test cases passed successfully, demonstrating that the registration module effectively handles user signup while maintaining data integrity and preventing duplicate accounts.

Table 4.2: Test Case for Login

ID	**Test Case Description**	**Test Data**	**Expected result**	**Actual Result**	**Test Result**
R_1	User enters valid login information	Email: rumba@gmail.com Password: Test@123	Redirects to HomePage	Redirects to HomePage	Pass
R_2	Users enters invalid login information	Email: rumba@gmail.com Password: asdfasdfasdf	Display Error Message	“Username or password is invalid”	Pass
R_3	User leaves the login fields empty	Email: rumba@gmail.com Password:	Display Error Message	“This field is required”.	Pass

In the above test case, the system's login functionality was thoroughly tested with three scenarios (L_1, L_2, and L_3). In L_1, a user successfully logged in with valid credentials, resulting in a seamless redirection to the homepage, as expected. L_2 examined the system's response to invalid login credentials, where it accurately displayed an error message prompting the user to enter correct information, thus maintaining security measures. Similarly, in L_3, the system appropriately detected empty login fields and prompted the user with an error message, ensuring comprehensive validation. All test cases passed, affirming the system's reliability in facilitating secure login procedures.

4.2.2 Test cases for System Testing

System testing involves verifying that the entire software system meets the specified requirements and functions correctly as a whole. This phase ensures that all integrated components work together seamlessly and deliver the expected outcomes.

Table 4.3: Test Case for System Testing

ID	Test Case Description	Test Data	Expected result	Actual result	Test Result
1	Job ID, User application details	Job ID, User application details	Job application is successfully submitted	As Expected	Pass
2	Company posts a new job	Job details (title, description, deadline, etc.)	Job is successfully posted and visible	As Expected	Pass
3.	Recruiter rejects a candidate	Candidate application details	Application status updated to "Rejected"	As Expected	Pass
4.	Job seeker views recommended jobs	User resume text	Relevant job listings are displayed	As Expected	Pass
5.	User searches for jobs	Search keywords	Relevant job listings matching keywords shown	As Expected	Pass

6.	User views their job application history	User ID	User’s past job applications are displayed	As Expected	Pass

In these test cases, key features of the job portal, including job application, job posting, candidate management, job recommendation, and job search, were thoroughly tested. All tests passed successfully, confirming that the system meets functional requirements and provides a seamless user experience for both job seekers and recruiters.

Chapter 5: Conclusion and Future Recommendation

5.1 Conclusion

The Job Portal system effectively streamlines the job search and recruitment process by providing personalized job recommendations based on user resumes using a Naive Bayes classification algorithm. This approach helps job seekers discover relevant job opportunities aligned with their skills and experience, while enabling recruiters to manage job postings and applications efficiently. The integration of AI-powered features such as smart job recommendations, industry insights, and customized quizzes enhances user engagement and supports informed career decisions. Overall, the system improves the match between candidates and jobs, reducing search time and increasing satisfaction for both job seekers and employers.

By automating classification and recommendation, the project addresses key challenges in online recruitment and delivers a scalable, user-centric platform for the evolving job market .

5.2 Future Recommendations

- Although this project successfully implements core functionalities and meets the main objectives, several enhancements can be incorporated in future iterations to further improve system performance and user experience:
  - Develop a more interactive and intuitive user interface for both recruiters and job seekers.
  - Integrate secure online payment gateways for premium job postings or subscription-based services.
  - Explore advanced machine learning and natural language processing techniques, such as deep learning models, to enhance job matching accuracy and handle more complex resume content.
  - Continuously collect and analyze user feedback to refine and personalize the recommendation engine dynamically.
  - Implement real-time updates to recommendations and application statuses to provide instant feedback to users.
  - Add modules for interview scheduling, automated resume parsing with enhanced PDF and document support, and recruiter analytics dashboards.

- Expand multi-language support and accessibility features to broaden the platform’s user base globally.

These future enhancements will strengthen the platform’s ability to connect job seekers and recruiters effectively, providing a seamless and personalized recruitment experience that adapts to the changing job market landscape.

References

[1] P. S. P. Vijayan and G. U. Ganapathy, "AI-based Resume Classification and Matching for Job Recommendation System," International Journal of Computer Applications, vol. 182, no. 19, pp. 1–5, 2021.

[2] M. A. Hossain, A. H. Hossain and M. Hasan, "Job Recommender System using Natural Language Processing and Machine Learning," 2020 International Conference on Computer, Communication, Chemical, Materials and Electronic Engineering (IC4ME2), pp. 1–4, doi:10.1109/IC4ME247184.2020.9038219.

[3] H. Patel and R. Shah, "Job Recommendation System Using Naive Bayes Algorithm," International Journal of Engineering Research & Technology (IJERT), vol. 9, no. 5, pp. 511–514, 2020.

[4] M. Gupta, "A Hybrid Approach for Career Path Recommendation Using Content Filtering and Quiz Assessment," M.Tech Thesis, Department of Computer Science, NIT Trichy, 2019.

[5] Y. Li and Q. Zhang, "AI-Based Job Market Analysis and Salary Prediction for Career Portals," *IEEE Access*, vol. 10, pp. 34671–34680, 2022.

[6] V. Thannimalai and L. Zhang, "A Content Based and Collaborative Filtering Recommender System," *2021 International Conference on Machine Learning and Cybernetics (ICMLC)*, 2021, pp. 1–7, doi:10.1109/ICMLC54886.2021.9737238.

[7] L. R. Roopesh and T. Bomatpalli, "A Survey of Travel Recommender System," *ResearchGate*, 2019, DOI:10.13140/RG.2.2.34775.32168.

[8] C. Srisawatsakul and W. Boontarig, "Tourism Recommender System using Machine Learning Based on User's Public Instagram Photos," *5th International Conference on Information Technology (InCIT)*, 2020, pp. 276–281, doi:10.1109/InCIT50588.2020.9317777.

[9] Rula M., "Smart Classification Framework for e-Tourism Data Management: A Review-Based Taxonomy," *International Journal of Computer Applications*, vol. 182, no. 19, pp. 32–38, 2021.

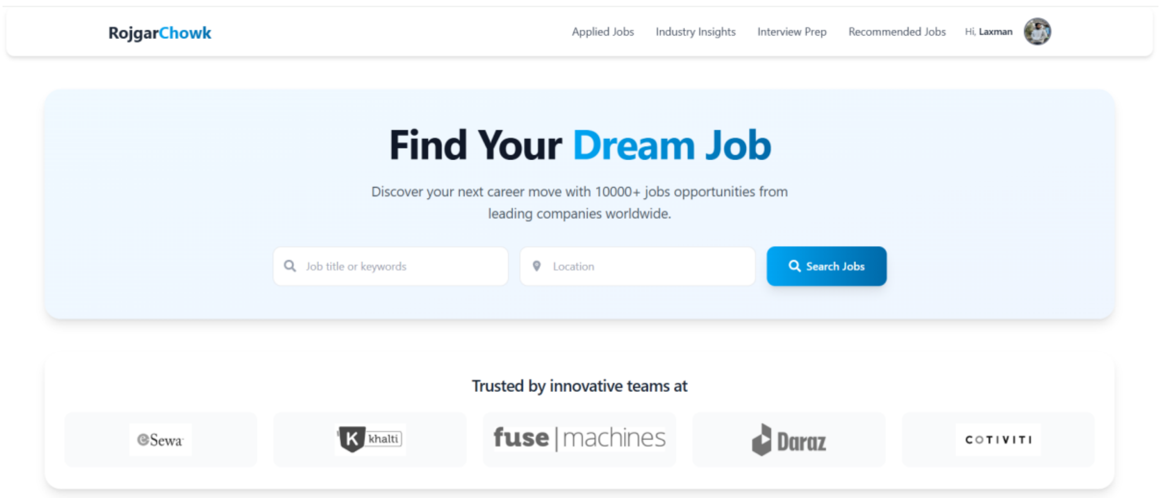
[10] A. Banerjee, "AI-based Decision Systems in Service Industries: A Live Prototype Case Study," *International Journal of Applied AI Research*, vol. 5, no. 2, pp. 145–152, 2020.

[11] Y. Zheng, "Entropy-Based Feature Selection for Travel Recommender Systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 5, pp. 1234–1245, 2021.

[12] P. Lops, M. De Gemmis, and G. Semeraro, "Content-based Recommender Systems: State of the Art and Trends," *Recommender Systems Handbook*, Springer, 2011, pp. 73–105.

APPENDICES

1. Home Page

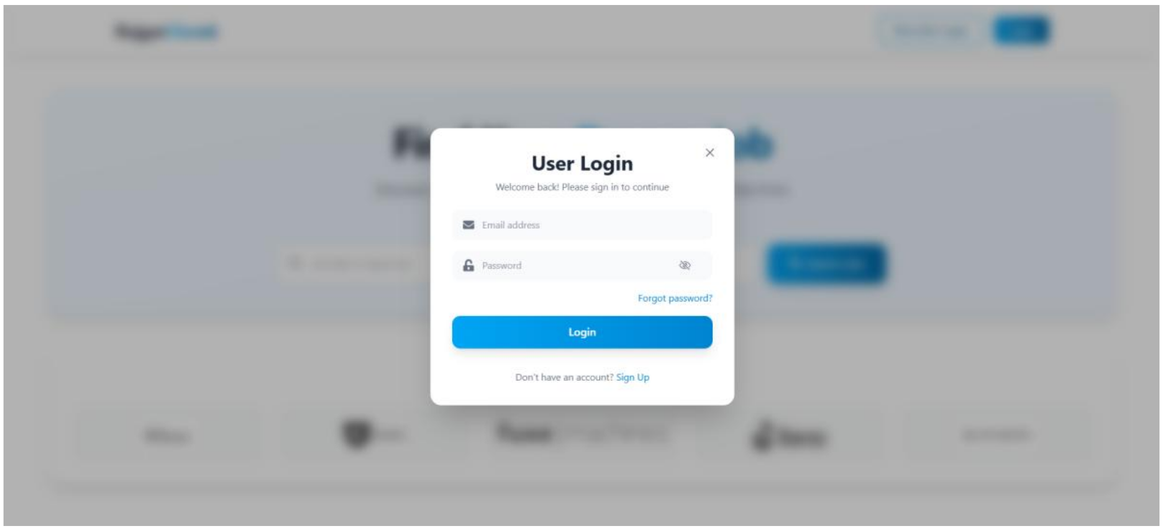


Find Your Dream Job

Discover your next career move with 10000+ jobs opportunities from leading companies worldwide.

Trusted by innovative teams at

2. Login Page

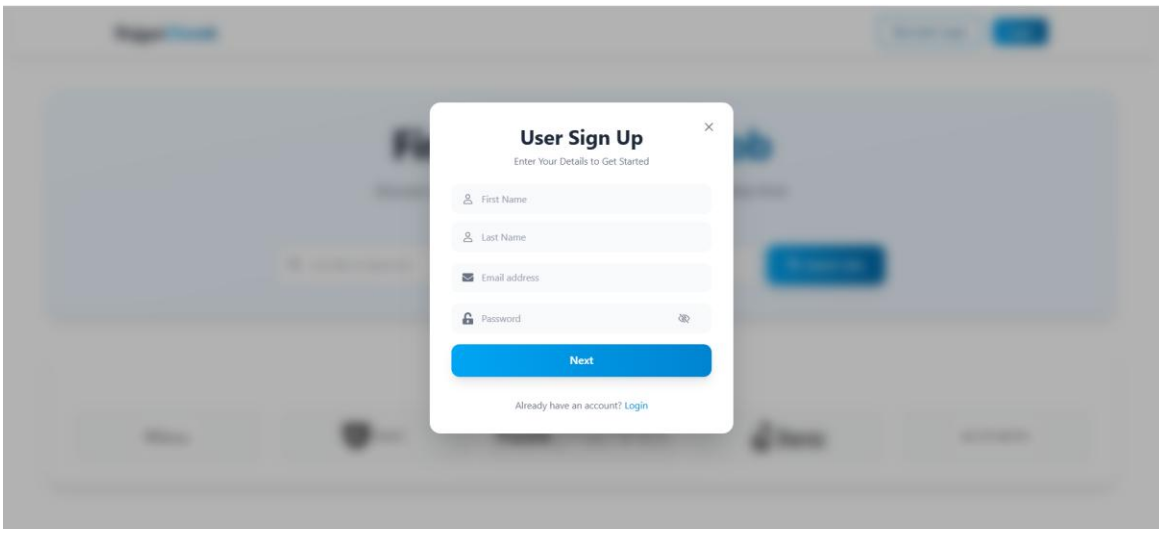


User Login

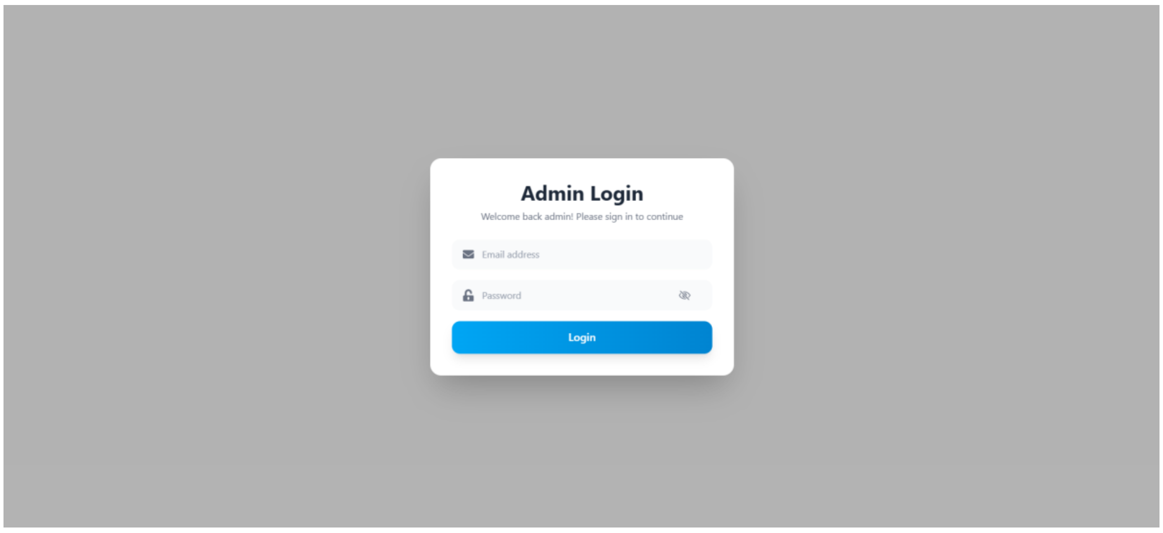
Welcome back! Please sign in to continue

Don't have an account? Sign Up

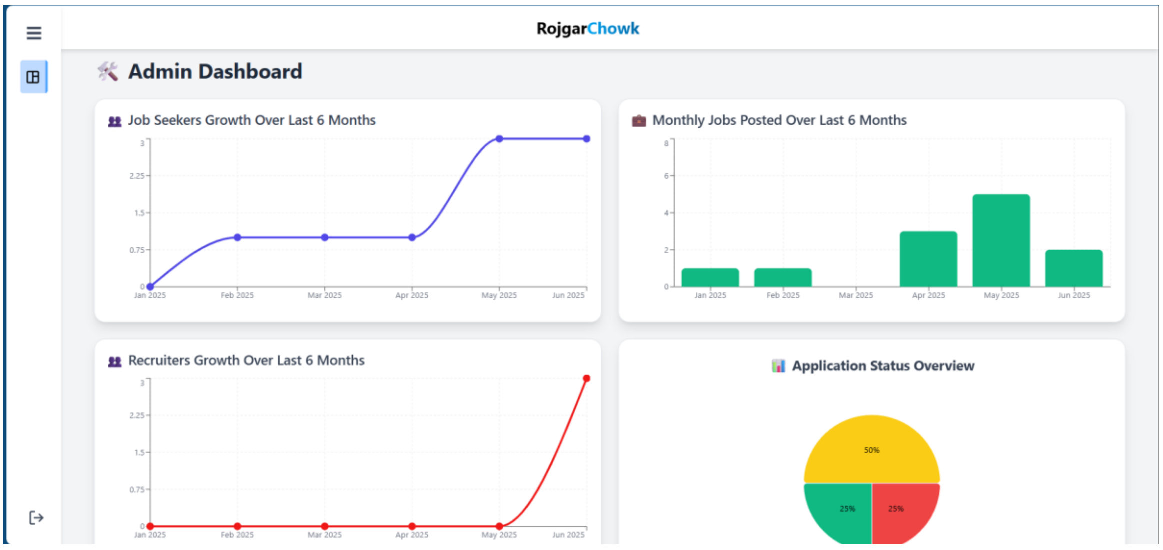
3. Register Page


User Sign Up
Enter Your Details to Get Started
First Name
Last Name
Email address
Password
Next
Already have an account? Login

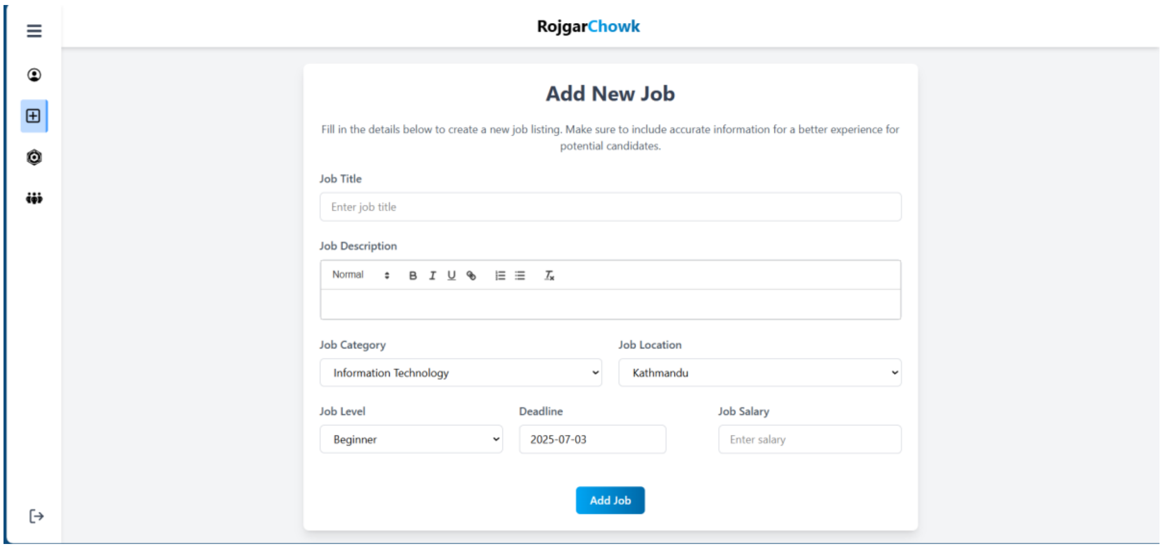
4. Admin Login Page



5. Admin Panel



6. Add New Job Page



Add New Job

Fill in the details below to create a new job listing. Make sure to include accurate information for a better experience for potential candidates.

Job Title: Enter job title

Job Description

Job Category: Information Technology

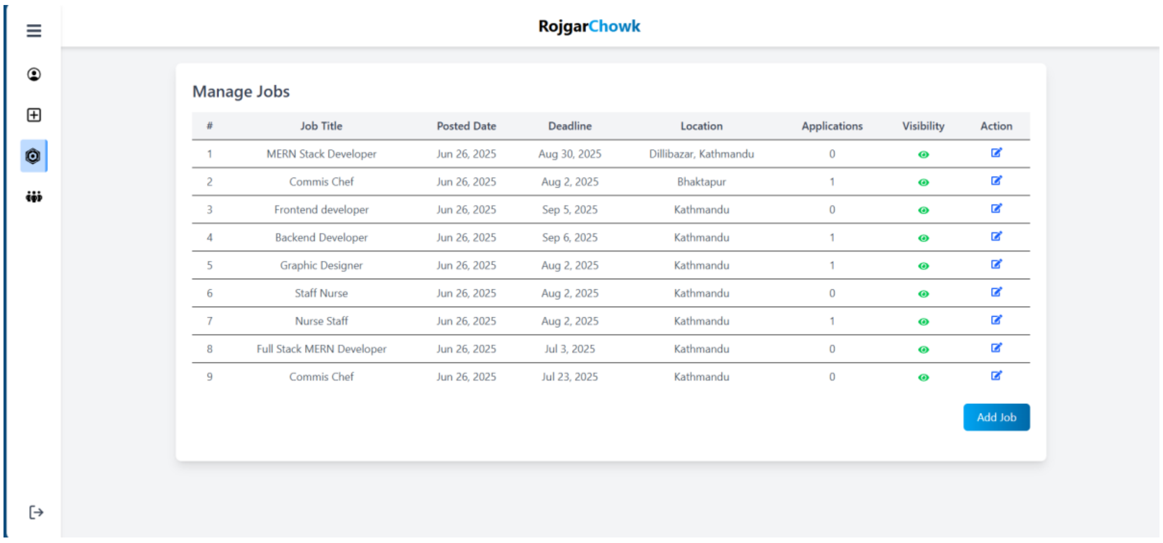
Job Location: Kathmandu

Job Level: Beginner

Deadline: 2025-07-03

Job Salary: Enter salary

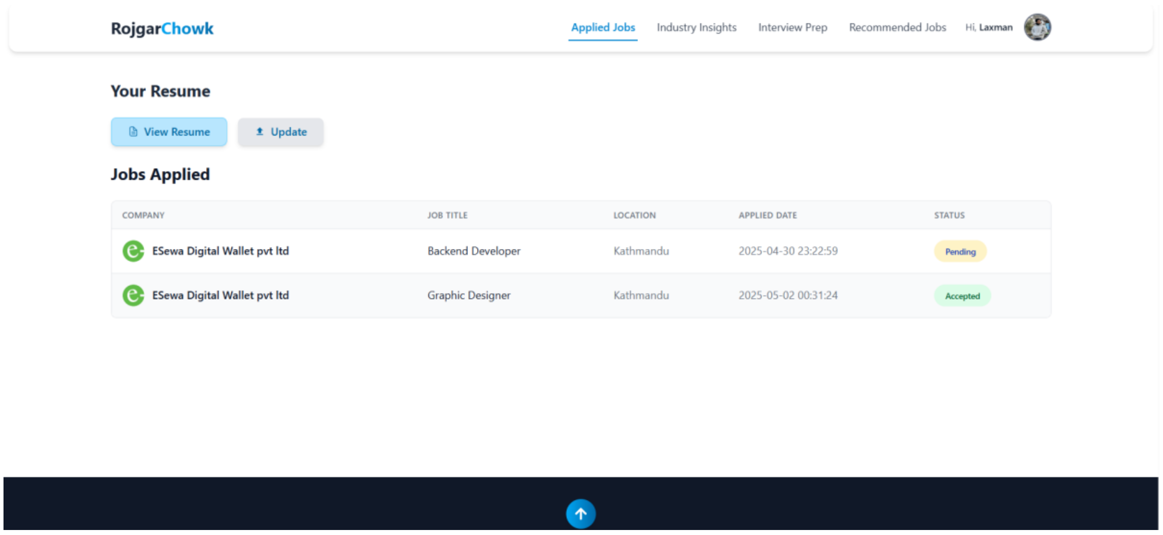
7. Manage Job Panel



Manage Jobs

#	Job Title	Posted Date	Deadline	Location	Applications	Visibility	Action
1	MERN Stack Developer	Jun 26, 2025	Aug 30, 2025	Dillibazar, Kathmandu	0	👁	✎
2	Commis Chef	Jun 26, 2025	Aug 2, 2025	Bhaktapur	1	👁	✎
3	Frontend developer	Jun 26, 2025	Sep 5, 2025	Kathmandu	0	👁	✎
4	Backend Developer	Jun 26, 2025	Sep 6, 2025	Kathmandu	1	👁	✎
5	Graphic Designer	Jun 26, 2025	Aug 2, 2025	Kathmandu	1	👁	✎
6	Staff Nurse	Jun 26, 2025	Aug 2, 2025	Kathmandu	0	👁	✎
7	Nurse Staff	Jun 26, 2025	Aug 2, 2025	Kathmandu	1	👁	✎
8	Full Stack MERN Developer	Jun 26, 2025	Jul 3, 2025	Kathmandu	0	👁	✎
9	Commis Chef	Jun 26, 2025	Jul 23, 2025	Kathmandu	0	👁	✎

8. User Applications Page

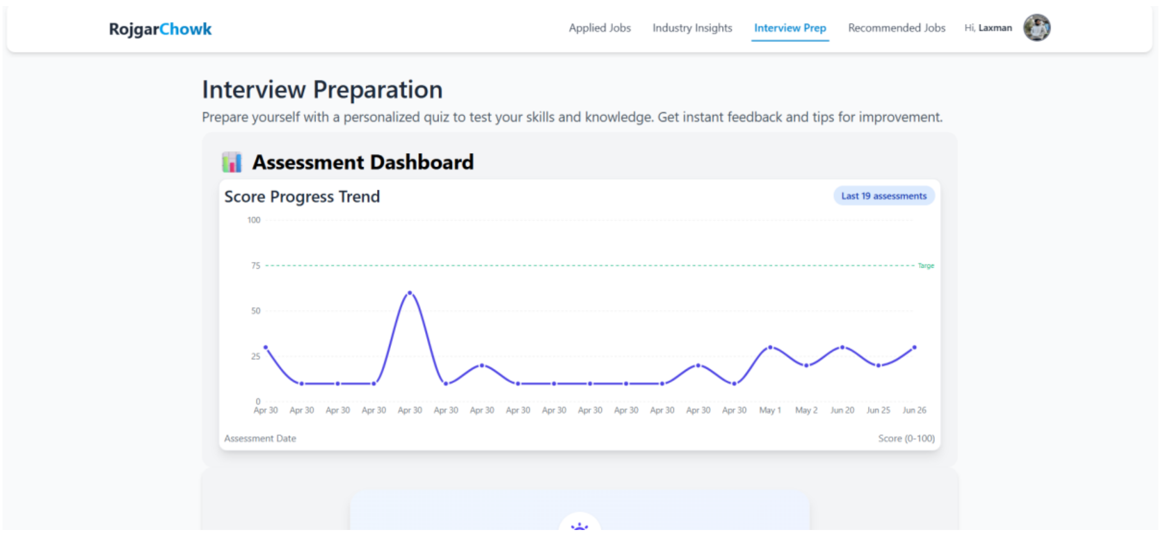


Your Resume

Jobs Applied

COMPANY	JOB TITLE	LOCATION	APPLIED DATE	STATUS
ESewa Digital Wallet pvt ltd	Backend Developer	Kathmandu	2025-04-30 23:22:59	Pending
ESewa Digital Wallet pvt ltd	Graphic Designer	Kathmandu	2025-05-02 00:31:24	Accepted

9. User Assessment Dashboard

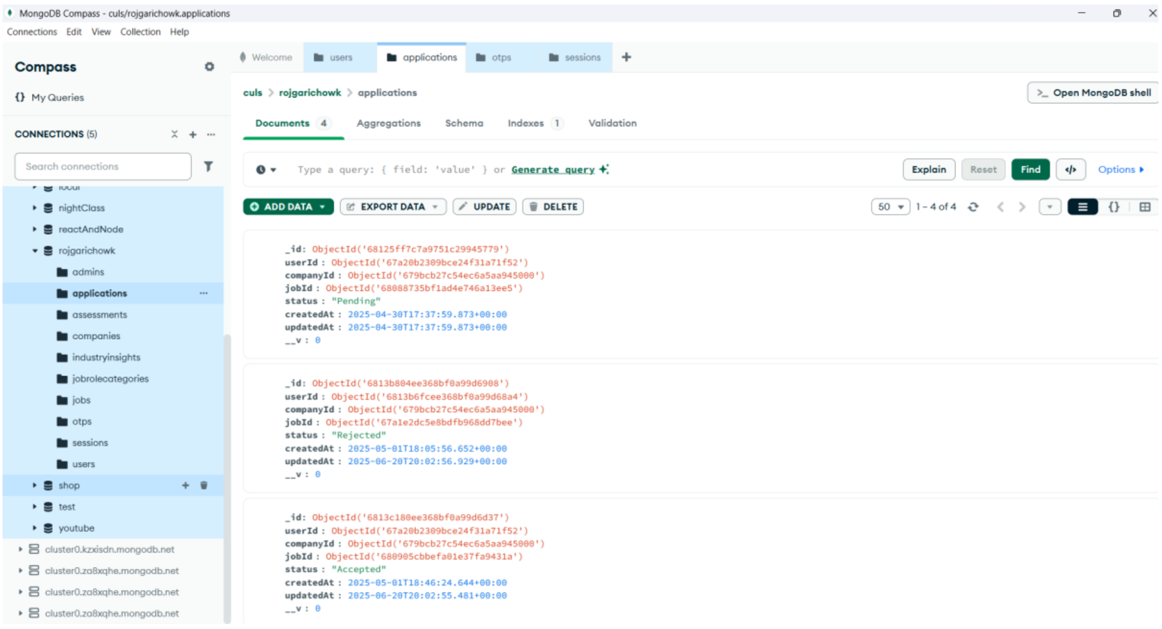


Interview Preparation

Prepare yourself with a personalized quiz to test your skills and knowledge. Get instant feedback and tips for improvement.

Assessment Dashboard

10.Database Overview



```
_id: ObjectId('68125ff7c7a9751c29945779')
userId : ObjectId('67a20b2309bce24f31a71f52')
companyId : ObjectId('679bcb27c54ec6a5aa945000')
jobId : ObjectId('68088735bf1ad4e746a13ee5')
status : "Pending"
createdAt : 2025-04-30T17:37:59.873+00:00
updatedAt : 2025-04-30T17:37:59.873+00:00
__v : 0

_id: ObjectId('6813b804ee368bf0a99d6908')
userId : ObjectId('6813b6fcee368bf0a99d68a4')
companyId : ObjectId('679bcb27c54ec6a5aa945000')
jobId : ObjectId('67a1e2dc5e8bdfb968dd7bee')
status : "Rejected"
createdAt : 2025-05-01T18:05:56.652+00:00
updatedAt : 2025-06-20T20:02:56.929+00:00
__v : 0

_id: ObjectId('6813c180ee368bf0a99d6d37')
userId : ObjectId('67a20b2309bce24f31a71f52')
companyId : ObjectId('679bcb27c54ec6a5aa945000')
jobId : ObjectId('680905cbbefa01e37fa9431a')
status : "Accepted"
createdAt : 2025-05-01T18:46:24.644+00:00
updatedAt : 2025-06-20T20:02:55.481+00:00
__v : 0
```